Theoretical Computer Science 410 (2009) 4554–4570

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Fixed-parameter algorithms for Kemeny rankings☆

Nadja Betzler[a,*], Michael R. Fellows[b], Jiong Guo[a], Rolf Niedermeier[a], Frances A. Rosamond[b]

[a] Institut für Informatik, Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 2, D-07743 Jena, Germany

[b] PC Research Unit, Office of DVC (Research), University of Newcastle, Callaghan, NSW 2308, Australia

ARTICLE INFO

Keywords:
Computational social choice
Voting systems
Winner determination
Rank aggregation
Consensus finding
Fixed-parameter tractability

ABSTRACT

The computation of Kemeny rankings is central to many applications in the context of rank aggregation. Given a set of permutations (votes) over a set of candidates, one searches for a "consensus permutation" that is "closest" to the given set of permutations. Unfortunately, the problem is NP-hard. We provide a broad study of the parameterized complexity for computing optimal Kemeny rankings. Besides the three obvious parameters "number of votes", "number of candidates", and solution size (called Kemeny score), we consider further structural parameterizations. More specifically, we show that the Kemeny score (and a corresponding Kemeny ranking) of an election can be computed efficiently whenever the *average* pairwise distance between two input votes is not too large. In other words, Kemeny Score is fixed-parameter tractable with respect to the parameter "average pairwise Kendall–Tau distance d_a". We describe a fixed-parameter algorithm with running time $16^{\lceil d_a \rceil} \cdot$ poly. Moreover, we extend our studies to the parameters "maximum range" and "average range" of positions a candidate takes in the input votes. Whereas Kemeny Score remains fixed-parameter tractable with respect to the parameter "maximum range", it becomes NP-complete in the case of an average range of two. This excludes fixed-parameter tractability with respect to the parameter "average range" unless P = NP. Finally, we extend some of our results to votes with ties and incomplete votes, where in both cases one no longer has permutations as input.

1. Introduction

To aggregate inconsistent information not only appears in classical voting scenarios but also in the context of meta search engines and a number of other application contexts [1,10,13,16,17]. In some sense, herein one deals with *consensus problems* where one wants to find a solution to various "input demands" such that these demands are met as well as possible. Naturally, contradicting demands cannot be fulfilled at the same time. Hence, the consensus solution has to provide a balance between opposing requirements. The concept of *Kemeny consensus* is among the most classical and important research topics in this context. In this paper, we study new algorithmic approaches based on parameterized complexity analysis [15,19,29] for computing Kemeny scores and, thus, optimal Kemeny rankings. We start with introducing Kemeny elections.

The Kemeny voting scheme goes back to the year 1959 [23] and was later specified by Levenglick [27]. It can be described as follows. An *election* (V, C) consists of a set V of n votes and a set C of m candidates. A vote is a *preference*

☆ This paper combines results from the two conference papers "Fixed-parameter algorithms for Kemeny scores" (*Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management (AAIM 2008)*, Springer, LNCS 5034, pages 60–71) and "How similarity helps to efficiently compute Kemeny rankings" (*Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*).

* Corresponding author. Tel.: +49 3641 9 46325.

E-mail addresses: nadja.betzler@uni-jena.de (N. Betzler), michael.fellows@newcastle.edu.au (M.R. Fellows), jiong.guo@uni-jena.de (J. Guo), rolf.niedermeier@uni-jena.de (R. Niedermeier), frances.rosamond@newcastle.edu.au (F.A. Rosamond).

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.08.033

list of the candidates, that is, each vote orders the candidates according to preference. For instance, in the case of three candidates a, b, c, the order $c > b > a$ would mean that candidate c is the best-liked one and candidate a is the least-liked one for this vote.[1] A "Kemeny consensus" is a preference list that is "closest" to the preference lists of the votes: For each pair of votes v, w, the so-called *Kendall–Tau distance* (*KT-distance* for short) between v and w, also known as the number of inversions between two permutations, is defined as

$$\text{dist}(v, w) = \sum_{\{c,d\}\subseteq C} d_{v,w}(c, d),$$

where the sum is taken over all unordered pairs $\{c, d\}$ of candidates, and $d_{v,w}(c, d)$ is set to 0 if v and w rank c and d in the same order, and is set to 1, otherwise. Using divide and conquer, the KT-distance can be computed in $O(m \cdot \log m)$ time (see, e.g., [25]). The *score* of a preference list l with respect to an election (V, C) is defined as $\sum_{v\in V} \text{dist}(l, v)$. A preference list l with the minimum score is called *Kemeny consensus* (or Kemeny ranking) of (V, C) and its score $\sum_{v\in V} \text{dist}(l, v)$ is the *Kemeny score* of (V, C). The central problem considered in this work is as follows:[2]

KEMENY SCORE
Input: An election (V, C) and a positive integer k.
Question: Is the Kemeny score of (V, C) at most k?

Clearly, in applications we are mostly interested in computing a Kemeny consensus of a given election. All our algorithms that decide the KEMENY SCORE problem actually provide a corresponding Kemeny consensus, which gives the desired ranking.

Known results. Bartholdi III et al. [2] showed that KEMENY SCORE is NP-complete, and it remains so even when restricted to instances with only four votes [16,17]. Given the computational hardness of KEMENY SCORE on the one hand and its practical relevance on the other hand, polynomial-time approximation algorithms have been studied. The Kemeny score can be approximated to a factor of 8/5 by a deterministic algorithm [34] and to a factor of 11/7 by a randomized algorithm [1]. Recently, a polynomial-time approximation scheme (PTAS) has been developed [24]. However, its running time is completely impractical. Conitzer, Davenport, and Kalagnanam [13,10] performed computational studies for the efficient exact computation of a Kemeny consensus, using heuristic approaches such as greedy and branch-and-bound. Their experimental results encourage the search for practically relevant, efficiently solvable special cases, thus motivating parameterized complexity studies. These experimental investigations focus on computing strong admissible bounds for speeding up search-based heuristic algorithms. In contrast, our focus is on exact algorithms with provable asymptotic running time bounds for the developed algorithms. Recently, a further experimental study for different approximation algorithms for Kemeny rankings has been presented [32]. Hemaspaandra et al. [21] provided further, exact classifications of the classical computational complexity of Kemeny elections. More specifically, whereas KEMENY SCORE is NP-complete, they provided $\mathbf{P}_{\|}^{\mathbf{NP}}$-completeness results for other, more general versions of the problem. Finally, it is interesting to note that Conitzer [9] uses a (different) notion of similarity (which is, furthermore, imposed on candidates rather than voters) to efficiently compute the closely related Slater rankings. Using the concept of similar candidates, he identifies efficiently solvable special cases, also yielding a powerful preprocessing technique for computing Slater rankings.

Our results. As pointed out by Conitzer et al. [10], for obvious reasons approximate solutions for election problems such as KEMENY SCORE may be of limited interest. Clearly, in political elections nobody would be satisfied with an approximate winner determination, and similar observations hold for other applications. Hence, *exact* solutions are of particular relevance in this context. Given the NP-completeness of the problem, however, it seems inevitable to live with exponential-time algorithms for solving KEMENY SCORE. Fortunately, parameterized complexity analysis as pioneered by Downey and Fellows [15,19,29] seems a fruitful approach here. Our results are summarized in Table 1. Studying the parameter "number of votes" is pointless because, as mentioned before, the problem is already NP-complete for only four votes. In contrast, by trying all possible permutations of the $m := |C|$ candidates, we can trivially attain an efficient algorithm if m is very small. The corresponding combinatorial explosion $m!$ in the parameter is fairly large, though. Using dynamic programming, we can improve this to an algorithm running in $O(2^m \cdot m^2 \cdot n)$ time where $n := |V|$.[3] To justify the practical relevance of this parameter, note that in political elections the number of candidates typically is much smaller than the number of votes. Our next parameterization uses the Kemeny score k as the parameter to derive an algorithm solving KEMENY SCORE in $O(1.53^k + m^2 n)$ time. This algorithm is based on a problem kernelization and a depth-bounded search tree. This may be considered as the "canonical parameterization" because the parameter measures the "solution size"—however, it is conceivable that in many applications the value of k may not be small, rendering the corresponding fixed-parameter algorithms impractical. Next, we introduce a structural parameterization by studying the parameter "maximum range $r_{\max}$ of positions that a candidate can have in any two of the input votes". We show that KEMENY SCORE can be solved in

[1] Some definitions also allow ties between candidates—we deal with this later in the paper.

[2] Note that for the sake of simplicity we formulate the decision problem here.

[3] Actually, basically the same result follows from an exact algorithm for the FEEDBACK ARC SET IN TOURNAMENTS problem due to Raman and Saurabh [31].

Table 1
Overview of the main results obtained in this work. To make the running times easier to read, we use the O^*-notation that allows us to omit polynomial factors.

	Kemeny Score	
Number of votes n [17]	NP-c for $n = 4$	
Number of candidates m	$O^*(2^m)$	(Section 5)
Kemeny score k	$O^*(1.53^k)$	(Section 4)
Maximum range of candidate positions r	$O^*(32^r)$	(Section 7)
Average range of candidate positions r_a	NP-c for $r_a \geq 2$	(Section 7)
Average KT-distance d_a	$O^*(16^{d_a})$	(Section 6)

$O(32^{r_{\max}} \cdot (r_{\max}^2 \cdot m + r_{\max} \cdot m^2 \log m \cdot n) + n^2 \cdot m \log m)$ time by a dynamic programming approach. Interestingly, if we study the parameter *average* range instead of *maximum* range, Kemeny Score turns NP-complete already for constant parameter values. By way of contrast, developing our main technical result, we again achieve fixed-parameter tractability when replacing the "range measure" by measuring the average pairwise distance between votes. More specifically, we show that with respect to the parameter "average KT-distance d_a" over all pairs of votes an exponential worst-case running time factor of 16^{d_a} is possible. This is practically meaningful because this parameterization shows that the consensus ranking can be computed efficiently whenever the votes are similar enough *on average*. Thus, it can cope with "outlier votes". Finally, we extend some of our findings to the cases where ties within votes are allowed and to incomplete votes where not all candidates are ranked, achieving fixed-parameter tractability as well as computational hardness results.

2. Preliminaries

We refer to the introductory section for some basic definitions concerning (Kemeny) elections. Almost all further concepts are introduced where needed. Here we restrict ourselves to some central concepts.

The following notion of "dirtiness" is of particular relevance for some of our algorithms.

Definition 1. Let (V, C) be an election. Two candidates $a, b \in C, a \neq b$, form a *dirty pair* if there exists one vote in V with $a > b$ and there exists another vote in V with $b > a$. A candidate is called *dirty* if he is part of a dirty pair, otherwise he is called *non-dirty*.[4]

Let the *position* of a candidate a in a vote v be the number of candidates that are better than a in v. That is, the leftmost (and best) candidate in v has position 0 and the rightmost has position $m - 1$. Then $\text{pos}_v(a)$ denotes the position of candidate a in v. For an election (V, C) and a candidate $c \in C$, the *average position* $p_a(c)$ of c is defined as

$$p_a(c) := \frac{1}{n} \cdot \sum_{v \in V} v(c).$$

For an election (V, C), the average KT-distance d_a is defined as[5]

$$d_a := \frac{1}{n(n-1)} \cdot \sum_{u,v \in V, u \neq v} \text{dist}(u, v).$$

Note that an equivalent definition is given by

$$d_a := \frac{1}{n(n-1)} \cdot \sum_{a,b \in C} \#v(a > b) \cdot \#v(b > a),$$

where for two candidates a and b the number of votes in which a is ranked better than b is denoted by $\#v(a > b)$. The latter definition is useful if the input is provided by the outcomes of the pairwise elections of the candidates including the margins of victory.

Furthermore, for an election (V, C) and for a candidate $c \in C$, the *range* $r(c)$ of c is defined as

$$r(c) := \max_{v,w \in V} \{|v(c) - w(c)|\} + 1.$$

The maximum range $r_{\max}$ of an election is given by $r_{\max} := \max_{c \in C} r(c)$ and the average range r_a is defined as

$$r_a := \frac{1}{m} \sum_{c \in C} r(c).$$

Finally, we briefly introduce the relevant notions of parameterized complexity theory [15,19,29]. Parameterized algorithmics aims at a multivariate complexity analysis of problems. This is done by studying relevant problem parameters

[4] For the sake of simplicity (and saving one letter), all candidates here are referred to by male sex.

[5] To simplify the presentation, the following definition counts the pair (u, v) *as well as* the pair (v, u), thus having to divide by $n(n-1)$ to obtain the correct average distance value.

$$
\begin{array}{lcl}
v_1 & : & a > b > c > d > e > f > \ldots \\
 & & \vdots \\
v_i & : & a > b > c > d > e > f > \ldots \\
v_{i+1} & : & b > a > d > c > f > e > \ldots \\
 & & \vdots \\
v_{2i} & : & b > a > d > c > f > e > \ldots
\end{array}
$$

Fig. 1. Small maximum range but large average KT-distance.

and their influence on the computational complexity of problems. The hope lies in accepting the seemingly inevitable combinatorial explosion for NP-hard problems, but confining it to the parameter. Thus, the decisive question is whether a given parameterized problem is *fixed-parameter tractable (FPT)* with respect to the parameter. In other words, for an input instance I together with the parameter k, we ask for the existence of a solving algorithm with running time $f(k) \cdot \text{poly}(|I|)$ for some computable function f.

A core tool in the development of fixed-parameter algorithms is polynomial-time preprocessing by *data reduction rules*, often yielding a *reduction to a problem kernel, kernelization* for short (also see [20] for a survey). Herein, the goal is, given any problem instance x with parameter k, to transform it in polynomial time into a new instance x' with parameter k' such that the size of x' is bounded from above by some function only depending on k, $k' \leq k$, and (x, k) is a yes-instance if and only if (x', k') is a yes-instance. We call a data reduction rule *sound* if the new instance after an application of this rule is a yes-instance iff the original instance is a yes-instance. We also employ search trees for our fixed-parameter algorithms. Search tree algorithms work in a recursive manner. The number of recursion calls is the number of nodes in the according tree. This number is governed by linear recurrences with constant coefficients. These can be solved by standard mathematical methods [29]. If the algorithm solves a problem instance of size s and calls itself recursively for problem instances of sizes $s - d_1, \ldots, s - d_i$, then $(d_1, \ldots, d_i)$ is called the *branching vector* of this recursion. It corresponds to the recurrence $T_s = T_{s-d_1} + \cdots + T_{s-d_i}$ for the asymptotic size T_s of the overall search tree.

3. On range and distance parameterizations of Kemeny score

This section discusses the "art" of finding different, practically relevant parameterizations of Kemeny Score.[6] Besides the considerations for the three obvious parameters "number of votes", "number of candidates", and "Kemeny score", our paper focusses on structural parameterizations, that is, structural properties of input instances that may be exploited to develop efficient solving algorithms for Kemeny Score. To this end, among others in this work, we investigate the realistic scenario (which, to some extent, is also motivated by previous experimental results [13,10]) that the given preference lists of the voters show some form of similarity. More specifically, we consider the parameters "average KT-distance" between the input votes, "maximum range of candidate positions", and "average range of candidate positions". Clearly, the maximum value is always an upper bound for the average value. The parameter "average KT-distance" reflects the situation that in an ideal world all votes would be the same, and differences occur due to some (limited) form of noise which makes the actual votes different from each other (see [14,12,11]). With average KT-distance as parameter, we can affirmatively answer the question whether a consensus list that is closest to the input votes can efficiently be found. By way of contrast, the parameterization by position range rather reflects the situation that whereas voters can be more or less decided concerning groups of candidates (e.g., political parties), they may be quite undecided and, thus, unpredictable, concerning the ranking within these groups. If these groups are small this can also imply small range values, thus making the quest for a fixed-parameter algorithm in terms of range parameterization attractive.

It is not hard to see, however, that the parameterizations by "average KT-distance" and by "range of position" can significantly differ. As described in the following, there are input instances of Kemeny Score that have a small range value and a large average KT-distance, and vice versa. This justifies separate investigations for both parameterizations; these are performed in Sections 6 and 7, respectively. We end this section with some concrete examples that exhibit the announced differences between our notions of vote similarity, that is, our parameters under investigation. First, we provide an example where one can observe a small maximum candidate range whereas one has a large average KT-distance, see Fig. 1. The election in Fig. 1 consists of $n = 2i$ votes such that there are two groups of i identical votes. The votes of the second group are obtained from the first group by swapping neighboring pairs of candidates. Clearly, the maximum range of candidates is 2. However, for m candidates the average KT-distance d_a is

$$d_a = \frac{2 \cdot (n/2)^2 \cdot (m/2)}{n(n-1)} > m/4$$

and, thus, d_a is unbounded for an unbounded number of candidates.

[6] Generally speaking, parameterized complexity analysis aims at "deconstructing intractability"; the corresponding systematic approach towards parameter identification has been recently discussed in more detail with respect to an NP-hard problem occurring in bioinformatics [26].

$$
\begin{array}{lllllllllllll}
v_1 & : & a & > & b & > & c & > & d & > & e & > & f & > & \ldots \\
v_2 & : & b & > & c & > & d & > & e & > & f & > & \ldots & > & a \\
v_1' & : & a & > & b & > & c & > & d & > & e & > & f & > & \ldots \\
 & & \vdots & & & & & & & & & & & &
\end{array}
$$

Fig. 2. Small average KT-distance but large maximum range.

Second, we present an example where the average KT-distance is small but the maximum range of candidates is large, see Fig. 2. In the election of Fig. 2 all votes are equal except that candidate a is at the last position in the second vote, but on the first position in all other votes. Thus, the maximum range equals the range of candidate a which equals the number of candidates, whereas by adding more copies of the first vote the average KT-distance can be made smaller than one.

Finally, we have a somewhat more complicated example displaying a case where one observes small average KT-distance but large average range of candidates.[7] To this end, we make use of the following construction based on an election with m candidates. Let V_m be a set of m votes such that every candidate is in one of the votes at the first and in one of the votes at the last position; the remaining positions can be filled arbitrarily. Then, for some $N > m^3$, add N further votes V_N in which all candidates have the same arbitrary order. Let $D(V_m)$ ($D(V_N)$) be the average KT-distance within the votes of V_m (V_N) and $D(V_N, V_m)$ be the average KT-distance between pairs of votes with one vote from V_N and the other vote from V_m. Since m^2 is an upper bound for the pairwise (and average) KT-distance between any two votes, it holds that $D(V_m) \leq m^2$, $D(V_N) = 1$, and $D(V_N, V_m) \leq m^2$. Moreover, we have $m \cdot (m-1)$ ordered pairs of votes within V_m, $N \cdot m$ pairs between V_N and V_m, and $N \cdot (N-1)$ pairs within V_N. Since $N > m^3$ it follows that

$$d_a \leq \frac{m(m-1) \cdot m^2 + Nm \cdot m^2 + N(N-1) \cdot 1}{N(N-1)} \leq 3.$$

In contrast, the range of every candidate is m, thus the average range is m.

4. Parameterization by the Kemeny score

In this section, we show that Kemeny Score is fixed-parameter tractable with respect to the Kemeny score k of the given election (V, C). More precisely, we present a kernelization and a search tree algorithm for this problem. The following natural lemma, whose correctness directly follows from the extended Condorcet criterion [33], is useful for deriving the problem kernel and the search tree.

Lemma 1. *Let a and b be two candidates in C. If $a > b$ in all votes $v \in V$, then every Kemeny consensus has $a > b$.*

4.1. Problem kernel

When applied to an input instance of Kemeny Score, the following polynomial-time executable data reduction rules yield an "equivalent" election with at most $2k$ candidates and at most $2k$ votes with k being the Kemeny score. In what follows, if we use a preference list over a *subset* of the candidates to describe a vote, then we mean that the remaining candidates are positioned arbitrarily in this vote. We first describe a reduction rule reducing the number of candidates and then we present a reduction rule reducing the number of votes.

The number of candidates. We apply the following data reduction rule to shrink the number of candidates in a given election (V, C).

Rule 1. Delete all candidates that are in no dirty pair.

Lemma 2. *Rule* 1 *is sound and can be carried out in $O(m^2 n)$ time.*

Proof. If a candidate c is not dirty, then there exists a partition of the remaining candidates into two subsets C_1 and C_2 such that in all votes all candidates $c_1 \in C_1$ are positioned better than c and all candidates $c_2 \in C_2$ are positioned worse than c. Thus, due to Lemma 1, the position of c in every Kemeny consensus is already determined and the removal of c does not affect the Kemeny score. The running time $O(m^2 n)$ of this rule is easy to see: For each candidate c, we construct an $n \times (m-1)$ binary matrix M. Each row corresponds to a vote and each column corresponds to a candidate $c' \neq c$. If c' has a better position than c in vote v, then the entry of row v and column c' of M has a value 1; otherwise, it is 0. Candidate c is a dirty candidate iff the rows of M are not identical. The matrix M can clearly be constructed in $O(mn)$ time. □

Lemma 3. *After having exhaustively applied Rule* 1*, in a yes-instance $((V, C), k)$ of* Kemeny Score *there are at most $2k$ candidates.*

[7] Clearly, this example also exhibits the situation of a large maximum candidate range with a small average KT-distance. We chose nevertheless to present the example from Fig. 2 because of its simplicity.

Proof. By Rule 1, we know that there are only dirty candidates in a reduced election instance. If there are more than $2k$ dirty candidates, then they must form more than k dirty pairs. For each dirty pair, there are two possibilities to position the candidates of this pair in any preference list. For both possibilities, the score of this preference list is increased by at least one, implying that, with more than k dirty pairs, there is no preference list with a score at most k. □

The number of votes. We apply a second simple data reduction rule to get rid of too many identical votes.

Rule 2. If there are more than k votes in V identical to a preference list l, then return "yes" if the score of l is at most k; otherwise, return "no".

In other words, the preference list l according to Rule 2 then is the only possible solution for the Kemeny consensus.

Lemma 4. *Rule* 2 *is sound and can be carried out in* $O(mn)$ *time.*

Proof. Regarding the soundness, assume that we have a Kemeny consensus that is not identical to l. Then, the KT-distance between l and this Kemeny consensus is at least 1. Since we have more than k copies of l the Kemeny score exceeds k, a contradiction. The running time can be obtained using the same matrix as in the proof of Lemma 2. □

The bound on the number of votes is achieved as follows.

Lemma 5. *After having exhaustively applied Rule* 1 *and Rule* 2*, in a yes-instance* $((V, C), k)$ *of* Kemeny Score *there are at most* $2k$ *votes.*

Proof. Between two distinct preference lists, the KT-distance is at least 1. By Rule 2, we know that a preference list has at most k "copies" in V. Therefore, if $|V| > 2k$, then the score of every preference list is at least $k + 1$ in (V, C). □

In summary, we achieve the following result.

Theorem 1. Kemeny Score *admits a problem kernel with at most* $2k$ *votes over at most* $2k$ *candidates. It can be computed in* $O(m^2n)$ *time.*

4.2. Search tree algorithm

It is easy to achieve an algorithm with search tree size $O(2^k)$ by branching on dirty pairs: At each search tree node we can branch into the two possible relative orders of a dirty pair and in each case we can decrease the parameter by one. For all non-dirty candidates their relative order with respect to all other candidates is already fixed due to Lemma 1. For the description of an improved search tree algorithm, we need the following definition.

Definition 2. A *dirty triple* consists of three candidates such that at least two pairs of them are dirty pairs.

The basic idea behind our algorithm is as follows. The search tree algorithm first enumerates all dirty pairs of the given election (V, C) and then branches according to the dirty triples. At a search tree node, in each case of the branching, an order of the candidates involved in the dirty triples processed at this node is fixed and maintained in a set. This order represents the relative positions of these candidates in the Kemeny consensus sought for. Then, the parameter is decreased according to this order. Since every order of two candidates in a dirty pair decreases the parameter at least by one, the height of the search tree is upper-bounded by the parameter.

Next, we describe the details of the branching strategy. At each node of the search tree, we store two types of information:

- The dirty pairs that have not been processed so far by ancestor nodes are stored in a set D.
- The information about the orders of candidates that have already been determined when reaching the node is stored in a set L. That is, for every pair of candidates whose order is already fixed we store this order in L.

For any pair of candidates a and b, the order $a > b$ is *implied* by L if there is a subset of ordered pairs $\{(a > c_1), (c_1 > c_2), (c_2 > c_3), \ldots, (c_{i-1} > c_i), (c_i > b)\}$ in L. To add the order of a "new" pair of candidates, for example $a > b$, to L, we must check if this is *consistent* with L, that is, L does not already imply $b > a$.

At the root of the search tree, D contains all dirty pairs occurring in (V, C). For all non-dirty pairs, their relative order in an optimal Kemeny ranking can be determined using Lemma 1. These orders are stored in L. At a search tree node, we distinguish three cases:

Case 1. If there is a dirty triple $\{a, b, c\}$ forming three dirty pairs contained in D, namely, $\{a, b\}, \{b, c\}, \{a, c\} \in D$, then remove these three pairs from D. Branch into all six possible orders of a, b, and c. In each subcase, if the corresponding order is not consistent with L, discard this subcase, otherwise, add the corresponding order to L and decrease the parameter according to this subcase. The branching vector of this case is (3, 4, 4, 5, 5, 6), giving the branching number 1.52. To see this, note that we only consider instances with at least three votes, since Kemeny Score is trivially solvable for only two votes. Thus, for every dirty pair $\{c, c'\}$, if there is only one vote with $c > c'$ (or $c' > c$), then there are at least two votes with $c' > c$ (or $c > c'$). A simple calculation then gives the branching vector. An example for a branching in this case is given in Fig. 3. For instance, the value 3 occurs because for the order $a > b > c$ we have three inversions, one with vote 2 and two with vote 3.

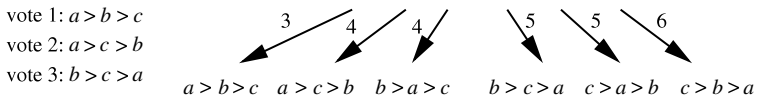

Fig. 3. An illustration of the branching for the dirty triple $\{a, b, c\}$ in the election given by votes 1, 2, and 3. For the six orders of the three candidates a, b, and c we can reduce the Kemeny score at the search tree node by the amount depicted next to the corresponding arrow.

Case 2. If Case 1 does not apply and there is a dirty triple $\{a, b, c\}$, then a, b, c form exactly two dirty pairs contained in D, say $\{a, b\} \in D$ and $\{b, c\} \in D$. Remove $\{a, b\}$ and $\{b, c\}$ from D. As $\{a, c\}$ is not a dirty pair, its order is determined by L. Hence, we have to distinguish the following two subcases.

If $a > c$ is in L, then branch into three further subcases, namely,

- $b > a > c$,
- $a > b > c$, and
- $a > c > b$.

For each of these subcases, we add the pairwise orders induced by them to L if they are consistent for all three pairs and discard the subcase, otherwise. The branching vector here is (3, 3, 2), giving the branching number 1.53.

If $c > a$ is in L, then we also get the branching vector (3, 3, 2) by branching into the following three further subcases:

- $b > c > a$,
- $c > b > a$, and
- $c > a > b$.

Case 3. If there is no dirty triple but at least one dirty pair (a, b) in D, then check whether there exists some relative order between a and b implied by L. If L implies no order between a and b, then do not branch: Instead, add an order between a and b to L that occurs in at least half of the given votes; otherwise, we add the implied order to L. Finally, decrease the parameter k according to the number of votes having a and b oppositely ordered compared to the order added to L.

The search tree algorithm outputs "yes" if it arrives at a node with $D = \emptyset$ and a parameter value $k \geq 0$; otherwise, it outputs "no". Observe that a Kemeny consensus is then the *full* order implied by L at a node with $D = \emptyset$.

Combining this search tree algorithm with the kernelization given in Section 4.1, we arrive at the main theorem of this section.

Theorem 2. Kemeny Score *can be solved in* $O(1.53^k + m^2 n)$ *time.*

Proof. First, we show that the above search tree algorithm solves Kemeny Score correctly. At the root of the search tree, we compute the set of dirty pairs, which is clearly doable in $O(m^2 n)$ time. Here, exemplarily we only give a correctness proof for Case 3. The other two cases can be shown in a similar but much simpler way. The branching vectors in these two cases are also easy to verify.

In Case 3, we reach a node v in the search tree with no dirty triple but at least one dirty pair $\{a, b\}$. If there is an order of a and b implied by L, then we have to obey this order, since it is fixed by v's ancestors in the search tree. Otherwise, without loss of generality, assume that at least half of the votes order a before b. It remains to show that, with the orders implied by L, there always exists a Kemeny consensus with $a > b$.

Consider a Kemeny consensus l obeying all orders from L and having $b > a$. Then, we switch the positions of a and b in l, getting another preference list l'. In the following, we show that the score $s(l')$ of l' is at most the score $s(l)$ of l. Since l and l' differ only in the positions of a and b, the difference between $s(l)$ and $s(l')$ can only come from the candidate pairs consisting of at least one of a and b. For the candidate pair $\{a, b\}$, switching the positions of a and b clearly does not increase $s(l')$ in comparison with $s(l)$. Here, we only consider the candidate pair $\{a, c\}$ with $c \neq b$. The case for the candidate pair $\{b, c\}$ can be handled similarly. If $c > b$ or $a > c$ in l, then the position switch does not affect $s(l) - s(l')$ with respect to c. Therefore, it suffices to show that there is no candidate $c \notin \{a, b\}$ such that l induces $b > c > a$. Due to Lemma 1, such a candidate c can exist only if there are some votes with $b > c$ and some votes with $c > a$. If all votes have $b > c$, then the order $b > c$ has already been added to the set L at the root of the search tree, since this order can be determined based on Lemma 1. If not all votes have $b > c$, then $\{b, c\}$ is a dirty pair. By the precondition of Case 3, $\{b, c\} \notin D$ and, thus, L contains the order $b > c$. Thus, in both cases, L contains $b > c$. Again, by the precondition of Case 3, (a, c) is not a dirty pair in D and, hence, there must be an order between a and c in L. The order $c > a$ cannot be in L; otherwise, $b > a$ would be implied by L, a contradiction. Further, $a > c$ in L contradicts the fact that l obeys the fixed orders in L. Therefore, there is no candidate c with $c \notin \{a, b\}$ such that l has $b > c > a$. This completes the proof of the correctness of Case 3 and the algorithm.

Concerning the running time, we use standard interleaving techniques [30,29] for kernelization and search trees, that is, at each node of the search tree we exhaustively apply our data reduction rules. In this way, we arrive at the running time of $O(1.53^k + m^2 n)$. □

The above search tree algorithm also works for instances in which the votes are weighted by positive integers. More specifically, one can use exactly the same search tree, but may gain some further (heuristic) speed-up by decreasing the parameter value according to the weights.

5. Parameterization by the number of candidates

As already mentioned in the introductory section, simply trying all permutations of candidates already leads to the fixed-parameter tractability of Kemeny Score with respect to the number m of candidates. However, the resulting algorithm has a running time of $O(m! \cdot nm \log m)$. Here, by means of dynamic programming, we improve this to an algorithm running in $O(2^m \cdot m^2 \cdot n)$ time.

Theorem 3. Kemeny Score *can be solved in* $O(2^m \cdot m^2 \cdot n)$ *time.*[8]

Proof. The dynamic programming algorithm goes like this: For each subset $C' \subseteq C$ compute the Kemeny score of the given election system (V, C) restricted to C'. The recurrence for a given subset C' is to consider every subset $C'' \subseteq C'$ where C'' is obtained by deleting a single candidate c from C'. Let l'' be a Kemeny consensus for the election system restricted to C''. Compute the score of the permutation l' of C' obtained from l'' by putting c in the first position. Take the minimum score over all l' obtained from subsets of C'. The correctness of this algorithm follows from the following claim.

Claim: A permutation l' of minimum score obtained as described above is a Kemeny consensus of the election system restricted to C'.

Proof of Claim: Let k' be a Kemeny consensus of the election system restricted to C', and suppose that candidate c is on the first position of k'. Consider $C'' = C' \setminus \{c\}$, and let k'' be the length-$|C''|$ tail of k'. The score of k' is $t + u$, where u is the score of k'' for the votes that are the restriction of V to C'', and t is the "cost" of putting c into the first position: Herein, the cost is the sum over the votes (restricted to C') of the distance of c from the first position in each vote. Now suppose that l'' is a Kemeny consensus of the election system restricted to C''. Compared with the score of k', augmenting l'' to l' by putting c in the first position increases the score of l' by exactly t. The score of l'' is at most u (since it is a Kemeny consensus for the election system restricted to C''), and so the score of l' is at most $t + u$, so it is a Kemeny consensus for the election system restricted to C'. This completes the proof of the claim.

For each of the subsets of C, the algorithm computes a Kemeny score. Herein, a minimum is taken from at most m values and the computation of each of these values needs $O(m \cdot n)$ time. As there are 2^m candidate subsets of C, the total running time is $O(2^m \cdot m^2 \cdot n)$. □

6. Parameterization by the average KT-distance

In this section, we further extend the range of parameterizations by giving a fixed-parameter algorithm with respect to the parameter "average KT-distance". We start with showing how the average KT-distance can be used to upper bound the range of positions that a candidate can take in any optimal Kemeny consensus. Based on this crucial observation, we then state the algorithm. Within this section, let $d := \lceil d_a \rceil$.

6.1. A crucial observation

Our fixed-parameter tractability result with respect to the average KT-distance of the votes is based on the following lemma.

Lemma 6. *Let d_a be the average KT-distance of an election (V, C) and $d = \lceil d_a \rceil$. Then, in every optimal Kemeny consensus l, for every candidate $c \in C$ with respect to its average position $p_a(c)$ we have $p_a(c) - d < l(c) < p_a(c) + d$.*

Proof. The proof is by contradiction and consists of two claims: First, we show that we can find a vote with Kemeny score less than $d \cdot n$, that is, the Kemeny score of the instance is less than $d \cdot n$. Second, we show that in every Kemeny consensus every candidate is in the claimed range. More specifically, we prove that every consensus in which the position of a candidate is not in a "range d of its average position" has a Kemeny score greater than $d \cdot n$, a contradiction to the first claim.

Claim 1: K-score$(V, C) < d \cdot n$.

Proof of Claim 1: To prove Claim 1, we show that there is a vote $v \in V$ with $\sum_{w \in V} \text{dist}(v, w) < d \cdot n$, implying this upper bound for an optimal Kemeny consensus as well. By definition,

$$d_a = \tfrac{1}{n(n-1)} \cdot \textstyle\sum_{v,w \in V, v \neq w} \text{dist}(v, w) \tag{1}$$

$$\Rightarrow \exists v \in V \text{ with } d_a \geq \tfrac{1}{n(n-1)} \cdot n \cdot \textstyle\sum_{w \in V, v \neq w} \text{dist}(v, w) \tag{2}$$

$$= \tfrac{1}{n-1} \cdot \textstyle\sum_{w \in V, v \neq w} \text{dist}(v, w) \tag{3}$$

$$\Rightarrow \exists v \in V \text{ with } d_a \cdot n > \textstyle\sum_{w \in V, v \neq w} \text{dist}(v, w). \tag{4}$$

[8] We give a direct proof here. Basically the same result also follows from a reduction to Feedback Arc Set on Tournaments [17] and a corresponding exact algorithm [31].

Since we have $d = \lceil d_a \rceil$, Claim 1 follows directly from Inequality (4).
The next claim shows the given bound on the range of possible candidates positions.

Claim 2: In every optimal Kemeny consensus l, every candidate $c \in C$ fulfills $p_a(c) - d < l(c) < p_a(c) + d$.

Proof of Claim 2: We start by showing that, for every candidate $c \in C$, we have

$$\text{K-score}(V, C) \geq \sum_{v \in V} |l(c) - v(c)|. \tag{5}$$

Note that, for every candidate $c \in C$, for two votes v, w we must have $\text{dist}(v, w) \geq |v(c) - w(c)|$. Without loss of generality, assume that $v(c) > w(c)$. Then, there must be at least $v(c) - w(c)$ candidates that have a smaller position than c in v and that have a greater position than c in w. Further, each of these candidates increases the value of $\text{dist}(v, w)$ by one. Based on this, Inequality (5) directly follows as, by definition, $K\text{-score}(V, C) = \sum_{v \in V} \text{dist}(v, l)$.

To simplify the proof of Claim 2, in the following, we shift the positions in l such that $l(c) = 0$. Accordingly, we shift the positions in all votes in V, that is, for every $v \in V$ and every $a \in C$, we decrease $v(a)$ by the original value of $l(c)$. Clearly, shifting all positions does not affect the relative differences of positions between two candidates. Then, let the set of votes in which c has a nonnegative position be V^+ and let V^- denote the remaining set of votes, that is, $V^- := V \setminus V^+$.

Now, we show that if candidate c is placed outside of the given range in an optimal Kemeny consensus l, then K-score$(V, C) > d \cdot n$. The proof is by contradiction. We distinguish two cases:

Case 1: $l(c) \geq p_a(c) + d$.

As $l(c) = 0$, in this case $p_a(c)$ becomes negative. Then,

$$0 \geq p_a(c) + d \Leftrightarrow -p_a(c) \geq d.$$

It follows that $|p_a(c)| \geq d$. The following shows that Claim 2 holds for this case.

$$\sum_{v \in V} |l(c) - v(c)| = \sum_{v \in V} |v(c)| \tag{6}$$

$$= \sum_{v \in V^+} |v(c)| + \sum_{v \in V^-} |v(c)|. \tag{7}$$

Next, replace the term $\sum_{v \in V^-} |v(c)|$ in (7) by an equivalent term that depends on $|p_a(c)|$ and $\sum_{v \in V^+} |v(c)|$. For this, use the following, derived from the definition of $p_a(c)$:

$$\begin{aligned} n \cdot p_a(c) &= \sum_{v \in V^+} |v(c)| - \sum_{v \in V^-} |v(c)| \\ \Leftrightarrow \sum_{v \in V^-} |v(c)| &= n \cdot (-p_a(c)) + \sum_{v \in V^+} |v(c)| \\ &= n \cdot |p_a(c)| + \sum_{v \in V^+} |v(c)|. \end{aligned}$$

The replacement results in

$$\begin{aligned} \sum_{v \in V} |l(c) - v(c)| &= 2 \cdot \sum_{v \in V^+} |v(c)| + n \cdot |p_a(c)| \\ &\geq n \cdot |p_a(c)| \geq n \cdot d. \end{aligned}$$

This says that K-score$(V, C) \geq n \cdot d$, a contradiction to Claim 1.

Case 2: $l(c) \leq p_a(c) - d$.

Since $l(c) = 0$, the condition is equivalent to $0 \leq p_a(c) - d \Leftrightarrow d \leq p_a(c)$, and we have that $p_a(c)$ is nonnegative. Now, we show that Claim 2 also holds for this case.

$$\begin{aligned} \sum_{v \in V} |l(c) - v(c)| &= \sum_{v \in V} |v(c)| = \sum_{v \in V^+} |v(c)| + \sum_{v \in V^-} |v(c)| \\ &\geq \sum_{v \in V^+} v(c) + \sum_{v \in V^-} v(c) = p_a(c) \cdot n \geq d \cdot n. \end{aligned}$$

Thus, also in this case, K-score$(V, C) \geq n \cdot d$, a contradiction to Claim 1. □

Based on Lemma 6, for every position we can define the set of candidates that can take this position in an optimal Kemeny consensus. The subsequent definition will be useful for the formulation of the algorithm.

Definition 3. Let (V, C) be an election. For every integer $i \in \{0, \ldots, m-1\}$, let P_i denote the set of candidates that can assume the position i in an optimal Kemeny consensus, that is, $P_i := \{c \in C \mid p_a(c) - d < i < p_a(c) + d\}$.

Using Lemma 6, we can easily show the following.

Lemma 7. *For every position i, $|P_i| \leq 4d$.*

Proof. The proof is by contradiction. Assume that there is a position i with $|P_i| > 4d$. Due to Lemma 6, for every candidate $c \in P_i$ the positions which c may assume in an optimal Kemeny consensus can differ by at most $2d - 1$. This is true because, otherwise, candidate c could not be in the given range around its average position. Then, in a Kemeny consensus, each of the at least $4d + 1$ candidates must hold a position that differs at most by $2d - 1$ from position i. As there are only $4d - 1$ such positions ($2d - 1$ on the left and $2d - 1$ on the right of i), one obtains a contradiction. □

6.2. Basic idea of the algorithm

In Section 6.4, we will present a dynamic programming algorithm for Kemeny Score. It exploits the fact that every candidate can only appear in a fixed range of positions in an optimal Kemeny consensus. The algorithm "generates" a Kemeny consensus from the left to the right. It tries out all possibilities for ordering the candidates locally and then combines these local solutions to yield an optimal Kemeny consensus. More specifically, according to Lemma 7, the number of candidates that can take a position i in an optimal Kemeny consensus for any $0 \leq i \leq m - 1$ is at most $4d$. Thus, for position i, we can test all possible candidates. Having chosen a candidate for position i, the remaining candidates that could also assume i must either be left or right of i in a Kemeny consensus. Thus, we test all possible two-partitionings of this subset of candidates and compute a "partial" Kemeny score for every possibility. For the computation of the partial Kemeny scores at position i we make use of the partial solutions computed for the position $i - 1$.

6.3. Definitions for the algorithm

To state the dynamic programming algorithm, we need some further definitions. For $i \in \{0, \ldots, m - 1\}$, let $I(i)$ denote the set of candidates that could be "inserted" at position i *for the first time*, that is,

$$I(i) := \{c \in C \mid c \in P_i \text{ and } c \notin P_{i-1}\}.$$

Let $F(i)$ denote the set of candidates that must be "forgotten" at latest at position i, that is,

$$F(i) := \{c \in C \mid c \notin P_i \text{ and } c \in P_{i-1}\}.$$

For our algorithm, it is essential to subdivide the overall Kemeny score into *partial Kemeny scores* (pK). More precisely, for a candidate c and a subset R of candidates with $c \notin R$, we set

$$\mathrm{pK}(c, R) := \sum_{c' \in R} \sum_{v \in V} d_v^R(c, c'),$$

where for $c \notin R$ and $c' \in R$ we have $d_v^R(c, c') := 0$ if in v we have $c > c'$, and $d_v^R(c, c') := 1$, otherwise. Intuitively, the partial Kemeny score denotes the score that is "induced" by candidate c and the candidate subset R if the candidates of R have greater positions than c in an optimal Kemeny consensus.[9] Then, for a Kemeny consensus $l := c_0 > c_1 > \cdots > c_{m-1}$, the overall Kemeny score can be expressed by partial Kemeny scores as follows.

$$\text{K-score}(V, C) = \sum_{i=0}^{m-2} \sum_{j=i+1}^{m-1} \sum_{v \in V} d_{v,l}(c_i, c_j) \tag{8}$$

$$= \sum_{i=0}^{m-2} \sum_{c' \in R} \sum_{v \in V} d_v^R(c_i, c') \text{ for } R := \{c_j \mid i < j < m\} \tag{9}$$

$$= \sum_{i=0}^{m-2} \mathrm{pK}(c_i, \{c_j \mid i < j < m\}). \tag{10}$$

Next, consider the corresponding three-dimensional dynamic programming table T. Roughly speaking, define an entry for every position i, every candidate c that can assume i, and every candidate subset $C' \subseteq P_i \backslash \{c\}$. The entry stores the "minimum partial Kemeny score" over all possible orders of the candidates of C' under the condition that c takes position i and all candidates of C' take positions smaller than i. To define the dynamic programming table formally, we need some further notation.

[9] By convention and somewhat counterintuitively, we say that a candidate c has a greater position than a candidate c' in a vote if $c' > c$.

Input: An election (V, C) and, for every $0 \leq i < m$, the set P_i of candidates that can assume position i in an optimal Kemeny consensus.

Output: The Kemeny score of (V, C).

Initialization:

01 **for** $i = 0, \ldots, m-1$
02 **for all** $c \in P_i$
03 **for all** $P_i' \subseteq P_i \backslash \{c\}$
04 $T(i, c, P_i') := +\infty$
05 **for all** $c \in P_0$
06 $T(0, c, \emptyset) := \mathrm{pK}(c, C \backslash \{c\})$

Update:

07 **for** $i = 1, \ldots, m-1$
08 **for all** $c \in P_i$
09 **for all** $P_i' \subseteq P_i \backslash \{c\}$
10 **if** $|P_i' \cup \bigcup_{j \leq i} F(j)| = i - 1$
and $T(i-1, c', (P_i' \cup F(i)) \backslash \{c'\})$ is defined **then**
11
$$T(i, c, P_i') = \min_{c' \in P_i' \cup F(i)} T(i-1, c', (P_i' \cup F(i)) \backslash \{c'\}) + \mathrm{pK}(c, (P_i \cup \bigcup_{i<j<m} I(j)) \backslash (P_i' \cup \{c\}))$$

Output:

12 K-score $= \min_{c \in P_{m-1}} T(m-1, c, P_{m-1} \backslash \{c\})$

Fig. 4. Dynamic programming algorithm for Kemeny Score exploiting the average KT-distance parameter.

Let $\Pi(C')$ denote the set of all possible orders of the candidates in C', where $C' \subseteq C$. Further, consider a Kemeny consensus in which every candidate of C' has a position smaller than every candidate in $C \backslash C'$. Then, the *minimum partial Kemeny score restricted to* C' is defined as

$$\min_{(d_1 > d_2 > \cdots > d_x) \in \Pi(C')} \left\{ \sum_{s=1}^{x} \mathrm{pK}\left(d_s, \{d_j \mid s < j < m\} \cup (C \backslash C')\right) \right\}$$

with $x := |C'|$. That is, it denotes the minimum partial Kemeny score over all orders of C'. We define an entry of the dynamic programming table T for a position i, a candidate $c \in P_i$, and a candidate subset $P_i' \subseteq P_i$ with $c \notin P_i'$. For this, we define $L := \bigcup_{j \leq i} F(j) \cup P_i'$. Then, an entry $T(i, c, P_i')$ denotes the minimum partial Kemeny score restricted to the candidates in $L \cup \{c\}$ under the assumptions that c is at position i in a Kemeny consensus, all candidates of L have positions smaller than i, and all other candidates have positions greater than i. That is, for $|L| = i - 1$, define

$$T(i, c, P_i') := \min_{(d_1 > \cdots > d_{i-1}) \in \Pi(L)} \left\{ \sum_{s=0}^{i-1} \mathrm{pK}(d_s, C \backslash \{d_j \mid j \leq s\}) \right\} + \mathrm{pK}(c, C \backslash (L \cup \{c\})).$$

6.4. Dynamic programming algorithm

The algorithm is displayed in Fig. 4. It is easy to modify the algorithm such that it outputs an optimal Kemeny consensus: for every entry $T(i, c, P_i')$, one additionally has to store a candidate c' that minimizes $T(i-1, c', (P_i' \cup F(i)) \backslash \{c'\})$ in line 11. Then, starting with a minimum entry for position $m-1$, one reconstructs an optimal Kemeny consensus by iteratively adding the "predecessor" candidate. The asymptotic running time remains unchanged. Moreover, in several applications, it is useful to compute not just *one* optimal Kemeny consensus but to enumerate all of them. At the expense of an increased running time, which clearly depends on the number of possible optimal consensus rankings, our algorithm can be extended to provide such an enumeration by storing all possible predecessor candidates.

Lemma 8. *The algorithm in Fig. 4 correctly computes* Kemeny Score.

Proof. For the correctness, we have to show two points:

First, all table entries are well defined, that is, for an entry $T(i, c, P_i')$ concerning position i there must be exactly $i-1$ candidates that have positions smaller than i. This condition is assured by line 10 of the algorithm.[10]

[10] It can still happen that a candidate takes a position outside of the required range around its average position. Since such an entry cannot lead to an optimal solution according to Lemma 6, this does not affect the correctness of the algorithm. To improve the running time it would be convenient to "cut away" such possibilities. We leave considerations in this direction to future work.

Second, we must ensure that our algorithm finds an optimal solution. Due to Equality (10), we know that the Kemeny score can be decomposed into partial Kemeny scores. Thus, it remains to show that the algorithm considers a decomposition that leads to an optimal solution. For every position, the algorithm tries all candidates in P_i. According to Lemma 6, one of these candidates must be the "correct" candidate c for this position. Further, for c we can observe that the algorithm tries a sufficient number of possibilities to partition all remaining candidates $C\backslash\{c\}$ such that they have either smaller or greater positions than i. More precisely, every candidate from $C\backslash\{c\}$ must be in exactly one of the following three subsets:

(1) The set F of candidates that have already been forgotten, that is, $F := \bigcup_{0\leq j\leq i} F(j)$.
(2) The set of candidates that can assume position i, that is, $P_i\backslash\{c\}$.
(3) The set I of candidates that are not inserted yet, that is, $I := \bigcup_{i<j<m} I(j)$.

Due to Lemma 6 and the definition of $F(j)$, we know that a candidate from F cannot take a position greater than $i-1$ in an optimal Kemeny consensus. Thus, it is sufficient to explore only those partitions in which the candidates from F have positions smaller than i. Analogously, one can argue that for all candidates in I, it is sufficient to consider partitions in which they have positions greater than i. Thus, it remains to try all possibilities for partitioning the candidates from P_i. This is done in line 09 of the algorithm. Thus, the algorithm returns an optimal Kemeny score. □

Theorem 4. Kemeny Score *can be solved in* $O(16^d \cdot (d^2 \cdot m + d \cdot m^2 \log m \cdot n) + n^2 \cdot m \log m)$ *time with average KT-distance* d_a *and* $d = \lceil d_a \rceil$. *The size of the dynamic programming table is* $O(16^d \cdot d \cdot m)$.

Proof. The dynamic programming procedure requires the set of candidates P_i for $0 \leq i < m$ as input. To determine P_i for all $0 \leq i < m$, one needs the average positions of all candidates and the average KT-distance d_a of (V, C). To determine d_a, compute the pairwise distances of all pairs of votes. As there are $O(n^2)$ pairs and the pairwise KT-distance can be computed in $O(m \log m)$ time [25], this takes $O(n^2 \cdot m \log m)$ time. The average positions of all candidates can be computed in $O(n \cdot m)$ time by iterating once over every vote and adding the position of every candidate to a counter variable for this candidate. Thus, the input for the dynamic programming algorithm can be computed in $O(n^2 \cdot m \log m)$ time.

Concerning the dynamic programming algorithm itself, due to Lemma 7, for $0 \leq i < m$, the size of P_i is upper-bounded by $4d$. Then, for the initialization as well as for the update, the algorithm iterates over m positions, $4d$ candidates, and 2^{4d} subsets of candidates. Whereas the initialization in the innermost instruction (line 04) can be done in constant time, in every innermost instruction of the update phase (line 11) one has to look for a minimum entry and one has to compute a pK-score. To find the minimum, one has to consider all candidates from $P'_i \cup F(i)$. As $P'_i \cup F(i)$ is a subset of P_{i-1}, it can contain at most $4d$ candidates. Further, the required pK-score can be computed in $O(n \cdot m \log m)$ time. Thus, for the dynamic programming we arrive at the running time of $O(m \cdot 4d \cdot 2^{4d} \cdot (4d + n \cdot m \log m)) = O(16^d \cdot (d^2 \cdot m + d \cdot m^2 \log m \cdot n))$.

Concerning the size of the dynamic programming table, there are m positions and any position can be assumed by at most $4d$ candidates. The number of considered subsets is bounded from above by 2^{4d}. Hence, the size of the table T is $O(16^d \cdot d \cdot m)$. □

7. Parameterization by the candidate range

In this section, we consider two further parameterizations, namely "maximum range" and "average range" of candidates. As exhibited in Section 3, the parameterizations by the maximum and average range in general are "orthogonal" to the parameterization by the average KT-distances dealt with in Section 6. Whereas for the parameter "maximum range" we can obtain fixed-parameter tractability by using the dynamic programming algorithm described in Section 6.4, Fig. 4, the Kemeny Score problem becomes NP-complete already in the case of an average range of two.

7.1. Parameter maximum range

In the following, we show how to bound the number of candidates that can assume a position in an optimal Kemeny consensus by a function of the maximum range. This enables the application of the algorithm from Fig. 4.

Lemma 9. *Let* $r_{\max}$ *be the maximum range of an election* (V, C). *Then, for every candidate its relative order in an optimal consensus with respect to all but at most* $3r_{\max}$ *candidates can be computed in* $O(n \cdot m^2)$ *time.*

Proof. According to Lemma 1, the following holds: If for two candidates $b, c \in C$ we have $v(b) > v(c)$ for all $v \in V$, then in every Kemeny consensus l it holds that $l(b) > l(c)$. Thus, it follows that for $b, c \in C$ with $\max_{v\in V} v(b) < \min_{v\in V} v(c)$, in an optimal Kemeny consensus l we have $l(b) < l(c)$. That is, for two candidates with "non-overlapping range" their relative order in an optimal Kemeny consensus can be determined using this observation. Clearly, all these candidate pairs can be computed in $O(n \cdot m^2)$ time.

Next, we show that for every candidate c there are at most $3r_{\max}$ candidates whose range overlaps with the range of c. The proof is by contradiction. Let the range of c go from position i to j, with $i < j$. Further, assume that there is a subset of candidates $S \subseteq C$ with $|S| \geq 3r_{\max} + 1$ such that for every candidate $s \in S$ there is a vote $v \in V$ with $i \leq v(s) \leq j$. Now, consider an arbitrary input vote $v \in V$. Since there are at most $3r_{\max}$ positions p with $i - r_{\max} \leq p \leq j + r_{\max}$ for one candidate $s \in S$ it must hold that $v(s) < i - r_{\max}$ or $v(s) > j + r_{\max}$. Thus, the range of s is greater than $r_{\max}$, a contradiction. Hence, there can be at most $3r_{\max}$ candidates that have a position in the range of c in a vote $v \in V$. As described above, for all other candidates we can compute the relative order in $O(n \cdot m^2)$ time. Hence, the lemma follows. □

As a direct consequence of Lemma 9, we conclude that every candidate can assume one of at most $3r_{\max}$ consecutive positions in an optimal Kemeny consensus. Recall that for a position i the set of candidates that can assume i in an optimal consensus is denoted by P_i (see Definition 3). Then, using the same argument as in Lemma 7, one obtains the following.

Lemma 10. *For every position i, $|P_i| \leq 6r_{\max}$.*

In complete analogy to Theorem 4, one arrives at the following.

Theorem 5. Kemeny Score *can be solved in $O(32^{r_{\max}} \cdot (r_{\max}^2 \cdot m + r_{\max} \cdot m^2 \log m \cdot n) + n^2 \cdot m \log m)$ time with maximum range $r_{\max}$. The size of the dynamic programming table is $O(32^{r_{\max}} \cdot r_{\max} \cdot m)$.*

7.2. Parameter average range

Theorem 6. Kemeny Score *is NP-complete for elections with average range two.*

Proof. The proof uses a reduction from an arbitrary instance $((V, C), k)$ of Kemeny Score to a Kemeny Score-instance $((V', C'), k)$ with average range less than two. The construction of the election (V', C') is given in the following. To this end, let a_i, $1 \leq i \leq |C|^2$, be new candidates not occurring in C.

- $C' := C \uplus \{a_i \mid 1 \leq i \leq |C|^2\}$.
- For every vote $v = c_1 > c_2 > \cdots > c_m$ in V, put the vote $v' := c_1 > c_2 > \cdots > c_m > a_1 > a_2 > \cdots > a_{m^2}$ into V'.

It follows from Lemma 1 that if a pair of candidates has the same order in all votes, it must have this order in a Kemeny consensus as well. Thus, in a Kemeny consensus it holds that $a_i > a_j$ for $i > j$ and, therefore, adding the candidates from $C' \backslash C$ does not increase the Kemeny score. Hence, an optimal Kemeny consensus of size k for (V', C') can be transformed into an optimal Kemeny consensus of size k for (V, C) by deleting the candidates of $C' \backslash C$. The average range of (V', C') is bounded as follows:

$$\begin{aligned} r_a &= \frac{1}{m + m^2} \cdot \sum_{c \in C'} r(c) \\ &= \frac{1}{m + m^2} \cdot \left(\sum_{c \in C} r(c) + \sum_{c \in C' \backslash C} r(c) \right) \\ &\leq \frac{1}{m + m^2} \cdot (m^2 + m^2) < 2. \end{aligned}$$

Clearly, the reduction can be easily modified to work for every constant value of at least two by choosing a C' of appropriate size. □

8. Ties and incomplete votes

In the following, we investigate two generalizations of Kemeny Score. First, we allow for ties, that is, in a vote several candidates may be ranked equally, and, second, we consider the realistic scenario that one has only incomplete information. Whereas all our results can be transferred to the problem variant with ties, the Kemeny score problem behaves in a significantly different way in the case of incomplete information. Note that the algorithm from Section 5 regarding the parameter "number of candidates" directly applies to both generalizations.

8.1. Kemeny score with ties

Introducing ties, we follow the definition of Hemaspaandra et al. [21]: Each vote consists of a *ranking* of candidates that can contain ties between candidates and the Kemeny consensus can contain ties as well. In the definition of the KT-distance, we then replace $d_{v,w}(c, d)$ by

$$t_{v,w}(c, d) := \begin{cases} 0 & \text{if } v \text{ and } w \text{ agree on } c \text{ and } d, \\ 1 & \text{if } v \text{ has a preference among } c \text{ and } d \text{ and } w \text{ does not,} \\ 2 & \text{if } v \text{ and } w \text{ strictly disagree on } c \text{ and } d. \end{cases}$$

In the following, we also work with weighted candidates, because we will use a dynamic programming algorithm that works on preprocessed instances with weighted candidates. Thus, in addition, we extend the KT-distance to weighted candidates.

Altogether, given a weight function $s : C \to N^+$, the KT-distance $\text{dist}(v, w)$ between two votes v, w is defined as

$$\text{dist}(v, w) = \sum_{\{c,d\} \subseteq C} s(c) \cdot s(d) \cdot t_{v,w}(c, d).$$

In what follows, we state our results for Kemeny Score with Ties.

8.1.1. Parameterization by the Kemeny score

Clearly, by increasing the Kemeny score by two if two votes strictly disagree on two candidates, the overall score becomes larger.

Theorem 7. Kemeny Score with Ties *admits a problem kernel with a linear number of votes and a linear number of candidates with respect to k. It can be solved in* $O(1.76^k + m^2n)$ *time.*

Proof. To prove the theorem, we extend the notion of dirty pairs such that two candidates a and b also form a dirty pair if we have $a = b$ in one vote and $a > b$ or $a < b$ in another vote. Then, we can obtain a problem kernelization in complete analogy to the case without ties (see Section 4). Concerning the search tree, by branching for a dirty pair $\{a, b\}$ into the three cases $a > b$, $a = b$, and $a < b$, we already achieve the branching vector (1, 2, 4) and, hence, the branching number 1.76. For example, having three votes in which candidates a and b tie in two of the votes and we have $a < b$ in the remaining vote, by branching into the case "$a = b$" we have one inversion that counts one, branching into "$a < b$" we have two inversions that count one, and branching into "$a > b$" we have two inversions counting one and one inversion counting two. Altogether, this gives the branching vector (1, 2, 4). □

8.1.2. Structural parameterizations

For Kemeny Score without ties, the algorithm used for showing fixed-parameter tractability for the average KT-distance strongly relies on the crucial observation that is described in Section 6.1. However, it is not obvious how to transfer this result to the case with ties. The same is true for the parameterization by the maximum range: Here it is not obvious that Lemma 9 can be transferred to the case with ties.

However, for the parameters "maximum range" and "maximum KT-distance", we still can obtain fixed-parameter tractability by using a dynamic programming given for the parameterization by the "maximum KT-distance" for the case without ties in our conference paper [3]. For the case without ties, this algorithm has a worse running time than the algorithm given in Section 6 and is omitted here. In contrast to the algorithm given in Section 6 that makes use of the fact that every candidate can only assume a certain range of positions in an optimal Kemeny consensus, the dynamic programming algorithm as given in our conference paper [3], does not need any "knowledge" about the consensus but explores the fact that every candidate only appears in a bounded range in all of the votes. Basically, the corresponding dynamic programming table stores all possible orders of the candidates of a given subset of candidates, resulting in a running time of $O^*((3d_{\max}+1)!)$ (or $O^*((3r_{\max}+1)!)$, respectively) [3].

Now, let us describe the results for ties. To deal with the two structural parameters "maximum range of candidate position" $r_{\max}$ and "maximum KT-distance" $d_{\max}$, we need to extend the notion of position and range.

For a vote $v \in V$ and a candidate $c \in C$, let

$$T_v(c) := \{c' \in C \mid c \text{ and } c' \text{ tie in } v\}.$$

Furthermore, for a vote v we define the minimum and maximum position which c can assume. That is, $\mathrm{pos}_v^{\min}(c)$ is the number of candidates that are better than c in v and $\mathrm{pos}_v^{\max}(c) := \mathrm{pos}_v^{\min}(c) + |T_v(c)|$. Then, we can define the range of a candidate as

$$r(c) := \max_{v,w \in V} \{|\mathrm{pos}_v^{\max}(c) - \mathrm{pos}_v^{\min}(c)|\}.$$

The overall maximum range of an election is defined as the maximum of all candidate ranges. Then, by arguing in analogy to our conference paper [3] we can state the following theorem.[11]

Theorem 8. Kemeny Score with Ties *can be solved in* $O((3r_{\max}+1)! \cdot 2^{3r_{\max}+1} \cdot r_{\max} \log r_{\max} \cdot nm)$ *time with* $r_{\max}$ *being the maximum position range of a candidate.*

With some more effort, we can also show that Kemeny Score with Ties is fixed-parameter tractable with respect to the parameter maximum KT-distance. To this end, we prove that the range of every candidate is at most two times the maximum KT-distance and then make use of Theorem 8. In order to achieve this, we employ a preprocessing step, which is based on the following lemma:

Lemma 11. *Let* $d_{\max}$ *denote the maximum KT-distance of an input instance. If there is a vote in which at least* $d_{\max}+2$ *candidates* $c_1, \ldots, c_{d_{\max}+2}$ *are tied, then the candidates* $c_1, \ldots, c_{d_{\max}+2}$ *are tied in all votes.*

Proof. We show the lemma by contradiction. Assume that $d_{\max}+2$ candidates $c_1 = c_2 = \cdots = c_{d_{\max}+2}$ are tied within a vote v. If there is a vote w in which not all candidates from $C' := \{c_1, \ldots, c_{d_{\max}+2}\}$ are tied, then there must be a non-empty subset $D \subsetneq C'$ containing candidates that are positioned in w better or worse than the remaining candidates $C' \setminus D$.

Then, to compute the KT-distance between v and w, for every candidate from D we have to increase the KT-distance by $x := |C'| - |D|$; in total, we have to increase the score by $|D| \cdot x$. It is easy to verify that the minimum value that $|D| \cdot x$ may take is $d_{\max}+1$, a contradiction to the fact that the KT-distance between two votes is at most $d_{\max}$. □

[11] The additional factor of $2^{3r_{\max}+1}$ in the running time is due to the fact that for every possible permutation of $3r+1$ candidates the algorithm checks all possibilities to set $=$ or $>$ between any two consecutive candidates in this permutation. This factor is missing in [3] due to a incorrect analysis of the running time.

Now, we can state the following data reduction rule:

Rule 3. If there are $d_{\max} + i$ candidates for any integer $i \geq 2$ that are tied in some vote, then replace them in all votes by one candidate whose weight is the sum of the weights of the replaced candidates.

The correctness of Rule 3 follows from Lemma 11 and it is easy to see that it can be carried out in $O(nm)$ time. Moreover, the dynamic programming procedure from our conference paper [3, Section 4] can be adapted in a straightforward way to deal with candidate weights. Note that, since Rule 3 can decrease the position range of a candidate, it also makes sense to combine Rule 3 with the dynamic programming algorithm exploiting this parameter.

Lemma 12. *In a candidate-weighted instance with maximum KT-distance $d_{\max}$ obtained after exhaustively applying Rule 3, the range of a candidate is at most $2d_{\max}$.*

Proof. Consider a candidate a and two votes v and w. Assume that $\text{pos}_v^{\min}(a) = p$ and $\text{pos}_w^{\max}(a) \geq p + 2d_{\max} + 1$. As after exhaustive application of Rule 3 in w at most $d_{\max}$ candidates can be tied with a, there are at least $p + d$ candidates that are better than a. In v there are at most $p - 1$ candidates that are better than a. Therefore, there are at least $d_{\max} + 1$ candidates that are ranked higher than a in w and lower than a in v. As a consequence, the KT-distance between v and w is at least $d_{\max} + 1$, a contradiction. □

Combining Lemma 12 and Theorem 8 one directly obtains the following.

Theorem 9. Kemeny Score with Ties *can be solved in $O((6d_{\max} + 2)! \cdot 2^{6d_{\max}+2} \cdot d_{\max} \cdot \log d_{\max} \cdot n \cdot m)$ time with d being the maximum KT-distance between two input votes.*

8.2. Incomplete votes

Finally, we consider the problem Kemeny Score with Incomplete Votes, which was introduced by Dwork et al. [16]. Here, the given votes are not required to be permutations of the entire candidate set, but only of candidate subsets, while the Kemeny consensus sought for should be a permutation of all candidates. In the definition of the KT-distance, we then replace $d_{v,w}(c, d)$ by

$$d'_{v,w}(c, d) := \begin{cases} 0 & \text{if } \{c, d\} \not\subseteq C_v \text{ or } \{c, d\} \not\subseteq C_w \text{ or } v \text{ and } w \text{ agree on } c \text{ and } d, \\ 1 & \text{otherwise,} \end{cases}$$

where C_v contains the candidates occurring in vote v.

For incomplete votes we cannot apply the kernelization and the branching approach of Section 4. This is due to the fact that we can have non-trivial instances without dirty pairs. An easy example is given by the following three votes over candidates a, b, and c:

- $v_1 : a > b$,
- $v_2 : b > c$,
- $v_3 : a > c$.

Here, although there is no dirty pair, a Kemeny consensus nevertheless is not obvious. However, using another approach, we can show that, parameterized by the Kemeny score k, also Kemeny Score with Incomplete Votes is fixed-parameter tractable.

Theorem 10. Kemeny Score with Incomplete Votes *is solvable in $O(nm^2 + 2^k \cdot nm^2 + k!4^k \cdot m^4k^3)$.*

Proof. Let $((V, C), k)$ be the given Kemeny Score instance. The fixed-parameter algorithm consists of three steps. First, transform the given election (V, C) system into one where each vote contains only two candidates. For example, a vote $a > b > c$ is replaced by threes votes: $a > b$, $b > c$, and $a > c$. Second, find all dirty pairs and, for each of them, branch into two cases such that in each case an order between the two candidates of this pair is fixed and the parameter k is decreased accordingly. Further, the votes corresponding to the dirty pair are removed. Third, after having processed all dirty pairs, we construct an edge-weighted directed graph for each of the election systems generated by the second step. The vertices of this graph one-to-one correspond to the candidates. There is an arc (c, c') from vertex c to vertex c' iff there is a vote of the form $c > c'$, and a weight equal to the number of the "$c > c'$"-votes is assigned to this arc. For each pair of candidates c and c' where the second step has already fixed an order $c > c'$, add an arc (c, c') and assign a weight equal to $k + 1$ to this arc. Observe that solving Kemeny Score on the instances generated by the first two steps is now equivalent to solving the Weighted Feedback Arc Set problems on the constructed edge-weighted directed graphs. Feedback Arc Set is the special case of Weighted Feedback Arc Set with unit edge weights. The fixed-parameter algorithm by Chen et al. [7] for Feedback Arc Set can also handle Weighted Feedback Arc Set with integer edge weights [6]. Therefore, applying this algorithm in the third step gives a running time of $O(k!4^k \cdot m^4k^3)$. Since the first step (transformation of the voting system) can be done in $O(m^2n)$ time and the branching is doable in $2^k \cdot nm^2$ time, we arrive at the claimed running time □

Table 2
Parameterized complexity of Kemeny Score and two of its generalizations. The NP-hardness results for a constant number of votes are from Dwork et al. [16, 17]. The remaining results are obtained in this work. All algorithms also provide a corresponding optimal Kemeny ranking. Note that within the table the maximum range (distance) is denoted by r (d) instead of $r_{\max}$ ($d_{\max}$).

	Kemeny Score	With ties	Incomplete votes
# votes n	NP-h for $n = 4$	NP-h for $n = 4$	NP-h for $n = 4$
# candidates m	$O^*(2^m)$	$O^*(2^m)$	$O^*(2^m)$
Kemeny score k	Kernelization $O^*(1.53^k)$	Kernelization $O^*(1.76^k)$	? $O^*(k! \cdot 4^k)$
Max. range r	$O^*(32^r)$	$O^*((3r+1)! \cdot 2^{3r+1})$	–
Avg. range r_a	NP-h for $r_a \geq 2$	NP-h for $r_a \geq 2$	–
Max. KT-dist. d	$O^*(16^d)$	$O^*((6d+2)! \cdot 2^{6d+2})$	NP-h for $d = 0$
Avg. KT-dist. d_a	$O^*(16^{d_a})$	?	NP-h for $d_a = 0$

In incomplete votes the position of a candidate in a vote cannot be defined. Therefore, we cannot parameterize by the maximum range of candidate positions.

Finally, we turn our attention to the parameter "maximum KT-distance". A simple reduction from the NP-complete Feedback Arc Set problem to Kemeny Score with Incomplete Votes shows that, in contrast to Kemeny Score with complete votes, the parameterization by the maximum KT-distance does not lead to fixed-parameter tractability for Kemeny Score with Incomplete Votes:

Theorem 11. Kemeny Score with Incomplete Votes *is NP-complete even if the maximum KT-distance between two input votes is zero.*

Proof. The problem is obviously contained in NP. To show the NP-hardness, we employ a straightforward reduction from Feedback Arc Set where, given a directed graph $G = (V, E)$ and a nonnegative integer k, one asks for an arc set $E' \subseteq E$ with $|E'| \leq k$ such that the removal of E' leaves an acyclic graph. This reduction is also described by Dwork et al. [16]. Let $(G = (V, E), k)$ be an instance of Feedback Arc Set. Set $C := V$ and, for each arc $(u, v) \in E$, add a vote $u > v$. Clearly, every feedback arc set E' with $|E'| \leq k$ of G corresponds to the existence of a Kemeny consensus having a score of k. Since the maximum KT-distance between two votes is zero in the constructed voting instance, we arrive at the claimed result. □

We remark that Theorem 11 clearly implies the NP-completeness for the average KT-distance parameterization as well.

9. Conclusion

We initiated a parameterized complexity analysis of Kemeny Score and some of its variants. An overview of our results is given in Table 2. Similar studies have been undertaken for Dodgson and Young elections in a companion paper [4]. Clearly, this work means only a first step. There are numerous possibilities for future studies. For instance, further (structural) parameterizations may be explored in order to extend the range of computationally tractable Kemeny Score instances. Clearly, it is also desirable to significantly improve on the running times of our fixed-parameter algorithms. For instance, one may try to refine the search tree strategy or provide further data reduction rules concerning the parameterization by the score value. In this context, above guarantee parameterization as introduced by Mahajan and Raman [28] seems to be a promising field of research. The point here is that $L := \sum_{\{a,b\} \subseteq C} \min\{\pi(a, b), \pi(b, a)\}$, where $\pi(a, b)$ denotes the number of votes in V that rank a higher than b, is an obvious lower bound for the Kemeny Score. Hence, it is interesting to parameterize above the guaranteed lower bound [28]. Also see Table 2 for few theoretical open questions, indicated by the question marks. Our positive algorithmic results encourage experimental studies for the developed algorithms to evaluate their practical usefulness. Here, perhaps synergies with the more heuristic approaches [10,13,32] may arise. Considering previous success stories with practical implementations of fixed-parameter algorithms [22], this line of research seems prospective. Notably, except for the parameter "Kemeny score", all other parameter values can be efficiently computed in advance from the input. Hence, implementing our algorithms, one can easily combine them into a "meta-algorithm" which first quickly computes all parameter values from the given input instance and then chooses which fixed-parameter algorithm(s) to apply (based on their predicted running time behavior, see Table 2). Finally, we want to advocate parameterized algorithmics [15,19,29] as a very helpful tool for better understanding and exploiting the numerous natural parameters occurring in voting scenarios with associated NP-hard combinatorial problems. Only few investigations in this direction have been performed so far, see, for instance [4,5,8,18].

Acknowledgements

We are grateful to anonymous referees of *AAIM 2008, COMSOC 2008, AAMAS 2009*, and *Theoretical Computer Science* whose constructive feedback significantly helped to improve the quality of the paper.

The first author was supported by the Deutsche Forschungsgemeinschaft, project PAWS (parameterized algorithmics for voting systems, NI 369/10) and project DARE (data reduction and problem kernels, GU 1023/1). The second author was

supported by the Australian Research Council. This work has been carried out while he was staying in Jena as a recipient of a Humboldt Research Award of the Alexander von Humboldt foundation, Bonn, Germany. The third author was partially supported by the Deutsche Forschungsgemeinschaft, Emmy Noether research group PIAF (fixed-parameter algorithms, NI 369/4). The fifth author was supported by the Australian Research Council.

References

[1] N. Ailon, M. Charikar, A. Newman, Aggregating inconsistent information: Ranking and clustering, Journal of the ACM 55 (5) (2008) Article 23 (October 2008).
[2] J. Bartholdi III, C.A. Tovey, M.A. Trick, Voting schemes for which it can be difficult to tell who won the election, Social Choice and Welfare 6 (1989) 157–165.
[3] N. Betzler, M.R. Fellows, J. Guo, R. Niedermeier, F.A. Rosamond, Fixed-parameter algorithms for Kemeny scores, in: Proc. of 4th AAIM, in: LNCS, vol. 5034, Springer, 2008, pp. 60–71.
[4] N. Betzler, J. Guo, R. Niedermeier, Parameterized computational complexity of Dodgson and Young elections, in: Proc. of 11th SWAT, in: LNCS, vol. 5124, Springer, 2008, pp. 402–413.
[5] N. Betzler, J. Uhlmann, Parameterized complexity of candidate control in elections and related digraph problems, Theoretical Computer Science, in press (doi:10.1016/j.tcs.2009.05.029), Available online 31 May 2009.
[6] J. Chen, Personal communication, December 2007.
[7] J. Chen, Y. Liu, S. Lu, B. O'Sullivan, I. Razgon, A fixed-parameter algorithm for the directed feedback vertex set problem, Journal of the ACM 55 (5) (2008) Article No. 21.
[8] R. Christian, M.R. Fellows, F.A. Rosamond, A. Slinko, On complexity of lobbying in multiple referenda, Review of Economic Design 11 (3) (2007) 217–224.
[9] V. Conitzer, Computing Slater rankings using similarities among candidates, in: Proc. of 21st AAAI, AAAI Press, 2006, pp. 613–619.
[10] V. Conitzer, A. Davenport, J. Kalagnanam, Improved bounds for computing Kemeny rankings, in: Proc. of 21st AAAI, AAAI Press, 2006, pp. 620–626.
[11] V. Conitzer, M. Rognlie, L. Xia, Preference functions that score rankings and maximun likelihood estimation, in: Proc. of 21st IJCAI, 2009, pp. 109–115.
[12] V. Conitzer, T. Sandholm, Common voting rules as maximum likelihood estimators, in: Proc. of 21st UAI, AUAI Press, 2005, pp. 145–152.
[13] A. Davenport, J. Kalagnanam, A computational study of the Kemeny rule for preference aggregation, in: Proc. of 19th AAAI, AAAI Press, 2004, pp. 697–702.
[14] M.J.A.N. de Caritat (Marquis de Condorcet), Essai sur l'application de l'analyse à la probabilité des décisions redues à la pluralité des voix, L'Imprimerie Royal, Paris, 1785.
[15] R.G. Downey, M.R. Fellows, Parameterized Complexity, Springer, 1999.
[16] C. Dwork, R. Kumar, M. Naor, D. Sivakumar, Rank aggregation methods for the Web, in: Proc. of 10th WWW, 2001, pp. 613–622.
[17] C. Dwork, R. Kumar, M. Naor, D. Sivakumar, Rank aggregation revisited, Manuscript, 2001.
[18] P. Faliszewski, E. Hemaspaandra, L.A. Hemaspaandra, J. Rothe, Llull and Copeland voting broadly resist bribery and control, in: Proc. of 22nd AAAI, AAAI Press, 2007, pp. 724–730.
[19] J. Flum, M. Grohe, Parameterized Complexity Theory, Springer, 2006.
[20] J. Guo, R. Niedermeier, Invitation to data reduction and problem kernelization, ACM SIGACT News 38 (1) (2007) 31–45.
[21] E. Hemaspaandra, H. Spakowski, J. Vogel, The complexity of Kemeny elections, Theoretical Computer Science 349 (2005) 382–391.
[22] F. Hüffner, R. Niedermeier, S. Wernicke, Techniques for practical fixed-parameter algorithms, The Computer Journal 51 (1) (2008) 7–25.
[23] J. Kemeny, Mathematics without numbers, Daedalus 88 (1959) 571–591.
[24] C. Kenyon-Mathieu, W. Schudy, How to rank with few errors, in: Proc. of 39th STOC, ACM, 2007, pp. 95–103.
[25] J. Kleinberg, É. Tardos, Algorithm Design, Addison Wesley, 2006.
[26] C. Komusiewicz, R. Niedermeier, J. Uhlmann, Deconstructing intractability—A case study for Interval Constrained Coloring, in: Proc. of 20th CPM, in: Lecture Notes in Computer Science, vol. 5577, Springer, 2009, pp. 207–220.
[27] A. Levenglick, Fair and reasonable election systems, Behavioral Science 20 (1) (1975) 34–46.
[28] M. Mahajan, V. Raman, Parameterizing about guaranteed values: MaxSat and MaxCut, Journal of Algorithms 31 (2) (1999) 335–354.
[29] R. Niedermeier, Invitation to Fixed-Parameter Algorithms, Oxford University Press, 2006.
[30] R. Niedermeier, P. Rossmanith, A general method to speed up fixed-parameter-tractable algorithms, Information Processing Letters 73 (2000) 125–129.
[31] V. Raman, S. Saurabh, Improved fixed parameter tractable algorithms for two edge problems: MAXCUT and MAXDAG, Information Processing Letters 104 (2) (2007) 65–72.
[32] F. Schalekamp, A. van Zuylen, Rank aggregation: Together we're strong, in: Proc. of 11th ALENEX, SIAM, 2009, pp. 38–51.
[33] M. Truchon, An extension of the Condorcet criterion and Kemeny orders. Technical Report, cahier 98-15 du Centre de Recherche en Économie et Finance Appliquées, Université Laval, Québec, Candada, 1998.
[34] A. van Zuylen, D.P. Williamson, Deterministic algorithms for rank aggregation and other ranking and clustering problems, in: Proc. of 5th WAOA, in: LNCS, vol. 4927, Springer, 2007, pp. 260–273.